مقدمةٌ عن البرمجة الكائنية OOP في إبداع (1)

الشرح التالي هو تعديلٌ غير كبير لمقالٍ قديمٍ شرحتُ فيه نمط البرمجة المسمي بالبرمجة المَقُودة بالكائنات object oriented programming أو اختصاراً (البرمجة الكائنية). و قد كنتُ أستخدم لغة الـ java في البداية لأشرح باستخدامها القواعد النظرية، و لكني قررتُ إعادة كتابة المقال مع استخدام لغة إبداع بدلاً من الـ java؛ لجعل هذا الشرح جزءٍ من محاولة تقريب قواعد إبداع للمبتدئين الذين لا تُناسبهم كتب المواصفات القياسية الخشنة الطابع.

***

ماهية البرمجة الكائنية:

لكي نفهم ماهية و خصائص نمط البرمجة الكائنية oop paradigm يجب علينا أن نتفهم المثال التالي:

فلنفترض أننا نريد كتابة برنامجٍ يقوم بإنشاء خَمس حساباتٍ لخمسة أشخاصٍ في خمس شركاتٍ مختلفة، و يقوم بحساب مستحقاتهم المالية لدي الشركة بعد مرور ثلاث سنوات بمعرفة نسبة الربح السنوي المعتادة من رأس المال. مع ملاحظة أن الكود سيكون بنمط البرمجة العادي (أي بدون استخدام الأصناف classes و الكائنات objects).

رقم حساب 1 = 100   مكسب 1 = 0.10
رقم حساب 2 = 150   مكسب 2 = 0.15
رقم حساب 3 = 175   مكسب 3 = 0.20
رقم حساب 4 = 200   مكسب 4 = 0.25
رقم حساب 5 = 250   مكسب 5 = 0.30

حساب 1 = بعد.سنوات(حساب 1 مكسب 1 3)
حساب 2 = بعد.سنوات(حساب 2 مكسب 2 3)
حساب 3 = بعد.سنوات(حساب 3 مكسب 3 3)
حساب 4 = بعد.سنوات(حساب 4 مكسب 4 3)
حساب 5 = بعد.سنوات(حساب 5 مكسب 5 3)

إجراء (رقم الحساب.الجديد) بعد.سنوات (رقم الحساب.القديم رقم المكسب رقم السنوات):
    رقم مميز = 1
    بينما مميز في من 1 إلي السنوات:
        الحساب.القديم = الحساب.القديم × المكسب + الحساب.القديم
    الحساب.الجديد = الحساب.القديم

بملاحظة البرنامج السابق سوف نري أننا لتمثيل كل حسابٍ من الحسابات المالية استخدمنا متغيرين من نوع رقم:

- الأول لتمثيل النقود التي يحتويها الحساب **حساب**،
- الثاني لتمثيل نسبة الربح **مكسب**،

و تم تكرار هذا مع كل حسابٍ من الحسابات المالية، بالطبع مع تغيير أسماء المتغيرات للتمييز بينها.
إذاً فالسؤال المنطقي الآن: هل يمكنني كمبرمجٍ أن أختصر كل ذلك الكود ما دام الكثير منه مكرراً بالفعل ؟

و الجواب: نعم، و هذا هو نمط البرمجة الكائنية.
فكل ما نفعله في البرمجة الكائنية أننا نضم الكود الذي يُشكل مع بعضه وحدةً بنائيةً متكاملة (سواء أكانت متغيرات أم إجراءات) و نضعها في مكانٍ واحدٍ و نُطلق عليها إسماً مُعيناً، ثم نكتفي بعد ذلك بأن نأمر جهاز الحاسب بصنع نسخةٍ من ذلك الكود و نُطلق علي تلك النسخة إسماً خاصاً بها.

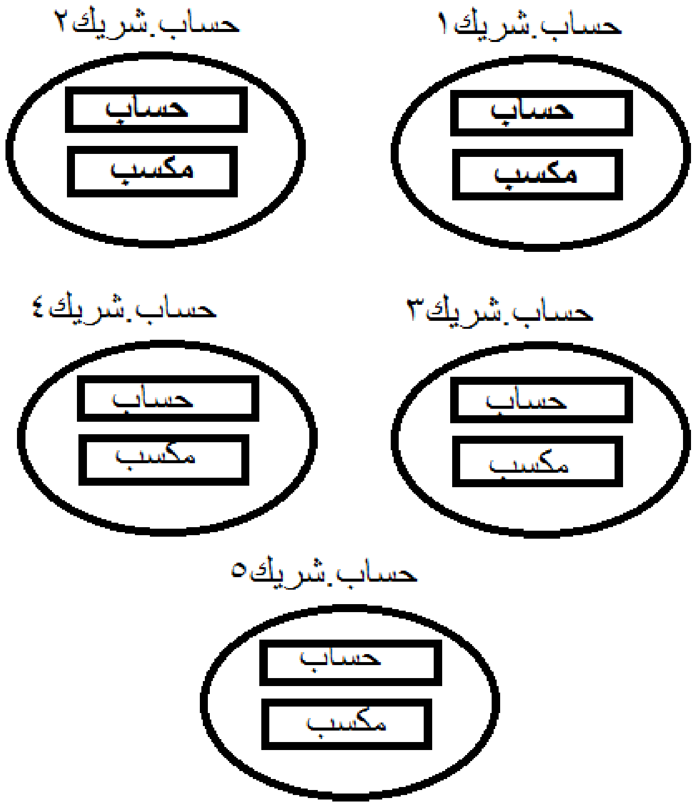
و بالتطبيق علي المثال السابق نري أن التقسيم الأولي لما سنحصل عليه سيكون كالتالي:

رقم حساب مكسب

إجراء (رقم الحساب.الجديد) بعد.سنوات (رقم السنوات):
رقم مميز = 1
بينما مميز في من 1 إلي السنوات:
حساب = حساب × مكسب + حساب
الحساب.الجديد = حساب

يمكن أن نُطلق علي الكود السابق إسم **حساب.شريك** و هو اسمٌ معبرٌ جداً عن وظيفته.
و في البرنامج الأساسي سنكتفي بطلب نسخ الكود حساب.شريك خمس مرات من مُترجِم اللغة بأسماء حساب.شريك 1، حساب.شريك 2، حساب.شريك 3، حساب.شريك 4، حساب.شريك 5.

فماذا سيفعل الحاسوب بعد حجز أماكن لتلك الكائنات في الذاكرة ؟
سوف يقوم بعمل ما يلي تقريباً:



و بداخل كل وحدةٍ من الوحدات المحجوزة تُوجد المتغيرات الموجودة في الكود المُكرر الذي كتبناه من قبل و وضعناه جانباً.

فلو كتبنا في البرنامج:

حساب.شركة 1_حساب

فإننا نعني المتغير **حساب** الذي هو من نوع رقم و المحجوز للنسخة المُسماة حساب.شركة 1 .

و هكذا مع:

حساب.شركة 2_حساب

و غيرها.

و لو أردنا أن ننفذ إجراءً علي النسخة **حساب.شركة 1** بحيث تعملُ باستخدام المتغيرات الخاصة بتلك النسخة و ليس الخاصة بأي نسخةٍ أخري فإننا نكتب الكود التالي:

حساب.شركة 1_اسم.الإجراء()

مثال:

حساب.شركة 1_أضيف.مبلغ(12345)

إلي الآن و الأمر مفهوم: يوجد لدينا كود مكرر في البرنامج فقمنا بكتابته مرةً واحدةً و أطلقنا عليه اسماً مُعيناً، ثم صنعنا منه ما نريد من نسخ.
و لكن هل هذا كل شيء ؟

لا.
فماذا لو أنني أردتُ إعطاء قيم ابتدائية للمتغيرات الموجودة داخل النسخ الجديدة، في نفس سطر تعريف كل نسخةٍ منها، كيف يمكنني فعل هذا ؟
و ماذا لو أنني أردتُ منع الوصول المباشر للمتغيرات داخل النسخ، مثل الأمر:

حساب.شركة 1_حساب = 10000

و أردتُ أن تكون هذه المتغيرات متاحةً فقط للإجراءات الموجودة معها في الكود المُكرّر فقط ؟
إلي آخر تلك الأسئلة التي تُوضح إجاباتُها خصائصَ نمط البرمجة الكائني.

سوف نقوم بتوضيح إجابات تلك الأسئلة فيما يلي، و لكن أولاً سوف نقوم بشرح المُصطلح العلمي المقابل لبعض الكلمات التي ذكرناها من قبل أثناء الشرح:

التوصيف	المُصطلح العلمي العربي	المُصطلح العلمي الإنقليزي
كود مكرر	صِنْف	class
نُسخة	نسخة، كائن	Object, instance
متغير داخل صنف	حقل بيانات	Data field
إجراء داخل صنف	إجراء حالة	Instance method

و لنُجِب الآن عن الأسئلة التي سبق و طرحناها.
أما كيفية إعطاء حقول البيانات قيماً ابتدائية في نفس جملة التعريف فيتم عن طريق ما يُسمي بالمُشَيِّد constructor، و هو إجراءٌ له نفس اسم الصنف class بدون أي نوعٍ من المُخرجات (ليس حتي void).

و ينقسم المُشَيِّد إلي نوعين:

- المُشَيِّد الافتراضي default constructor
- المُشَيِّد ذو المُدخَلات parameterized constructor

و هذا هو المثال السابق بالكامل (مع إضافة مُشَيِّداتٍ من ذوات المُدخَلات):

حساب.شريك حساب.شريك 1 = حساب.شريك(100 0.10)
حساب.شريك حساب.شريك 2 = حساب.شريك(150 0.15)
حساب.شريك حساب.شريك 3 = حساب.شريك(175 0.20)
حساب.شريك حساب.شريك 4 = حساب.شريك(200 0.25)
حساب.شريك حساب.شريك 5 = حساب.شريك(250 0.30)

صنف حساب.شريك :
رقم حساب مكسب

إجراء حساب.شريك(رقم حساب رقم مكسب):
هذا_حساب = حساب
هذا_مكسب = مكسب

إجراء (رقم الحساب.الجديد) بعد.سنوات (رقم السنوات):
رقم مميز = 1
بينما مميز في من 1 إلي السنوات:
حساب = حساب × مكسب + حساب
الحساب.الجديد = حساب

و يقوم البرنامج بحجز مكانٍ في الذاكرة للكائن الجديد ثم تنفيذ المُشَيِّد الافتراضي عندما نكتب ما يلي:

حساب.شريك حساب.شريك 1 = حساب.شريك()

و لاحظ عدم وجود قيم بين قوسي المُشَيِّد المُستدعَى.
بينما يقوم البرنامج بحجز مكان الذاكرة ثم تنفيذ المُشَيِّد ذي المُدخلات لو كتبنا ما يلي:

حساب.شريك حساب.شريك 1 = حساب.شريك(100 0.10)

و لاحظ وجود قيمٍ بين قوسي المُشَيِّد المُستدعَى. و في هذه الحالة يكون:

حساب.شريك 1_حساب = 100
حساب.شريك 1_مكسب = 0.10

أما منع الوصول المُباشر لحقول البيانات فيكون ببساطة بإعطائها الخاصية خاص عند تعريفها، مثل:

رقم خاص حساب

و لو أننا قمنا بتعريف كل حقول البيانات الموجودة في الصنف و أعطيناها الخاصية خاص فسنجد أننا وصلنا إلي جعل الصنف أشبه بالكبسولة؛ حيث لا يُمكن الوصول لحقول البيانات إلا من خلال الإجراءات الموجودة في الصنف، و هو ما يُعرف بالكبسلة encapsulation.

و كذلك ينطبق مفهوم الكبسلة علي كود الإجراءات الموجودة داخل الصنف، حيث لا يُمكن للمستخدم في البرنامج النهائي أن يُغير من ذلك الكود، بل هو مجرد مستخدمٍ له فقط.

***

إلي هنا نكون قد أنهينا المفاهيم الأساسية في عالم البرمجة الكائنية، و لكن أشياء كثيرة للغاية ما زالت تنتظر و سنقوم بشرحها في مقالاتٍ قادمةٍ بإذن الله تعالي.

***

لمعرفة المزيد عن المشروع يمكنكم زيارة الموقع الرسمي له:
http://ebda3lang.blogspot.com